CECS 528, Exam 2, Spring 2026, Dr. Ebert

IMPORTANT: READ THE FOLLOWING DIRECTIONS. Directions,

- For each LO and Additional problem, write your solutions to all parts using **ONE SHEET OF PAPER ONLY (BOTH FRONT AND BACK)**. Write **NAME** and **PROBLEM NUMBER** on each sheet.
- Use **SEPARATE SHEETS** for different LO/Additional problems.
- Make up LO solutions may be written on the same sheet.

Unit 2 LO Problems

LO5. Do the following.

(a) The dynamic-programming algorithm that solves the `Optimal Binary Search Tree` optimization problem defines a recurrence for the function $\text{wac}(i, j)$. State in words the meaning of $\text{wac}(i, j)$ and provide its recurrence. (7 pts)

Solution. See lecture notes.

(b) Provide the recurrence for the `0-1 Knapsack` optimization problem and use it to compute the maximum profit that is obtainable for an instance that has following items and capacity $M = 8$. (9 pts)

Item	a	b	c	d	e	f
Weight	3	2	1	1	2	2
Profit	50	30	25	30	10	35

Sum to 145

Solution. See notes for recurrence.

P(i,c)

i \ c	0	1	2	3	4	5	6	7	8
0	0	0	0	0	0	0	0	0	0
a' 1	0	0	0	50	50	50	50	50	50
b 2	0	0	30	50	50	80	80	80	80
c 3	0	25	30	55	75	80	105	105	105
d 4	0	30	55	60	85	105	110	135	135
e 5	0	30	55	60	85	105	110	135	135
f 6	0	30	55	65	90	105	120	140	145

(c) The following is the dynamic-programming table for an instance of `Edit Distance`. The first column displays the letters of word u. Use the table to determine a v word that is consistent with the table and provide the sequence of right-to-left edits that transform u to v. Assume that v only uses the letters `a` and `b`. **Note: correctly solving this problem is unnecessary for passing LO5**. (9 pts)

u_i/v_j	λ								
λ	0	1	2	3	4	5	6	7	8
a	1	1	1	2	3	4	5	6	7
b	2	1	2	1	2	3	4	5	6
b	3	2	2	2	1	2	3	4	5
a	4	3	2	3	2	2	3	4	5
a	5	4	3	3	3	3	3	4	5

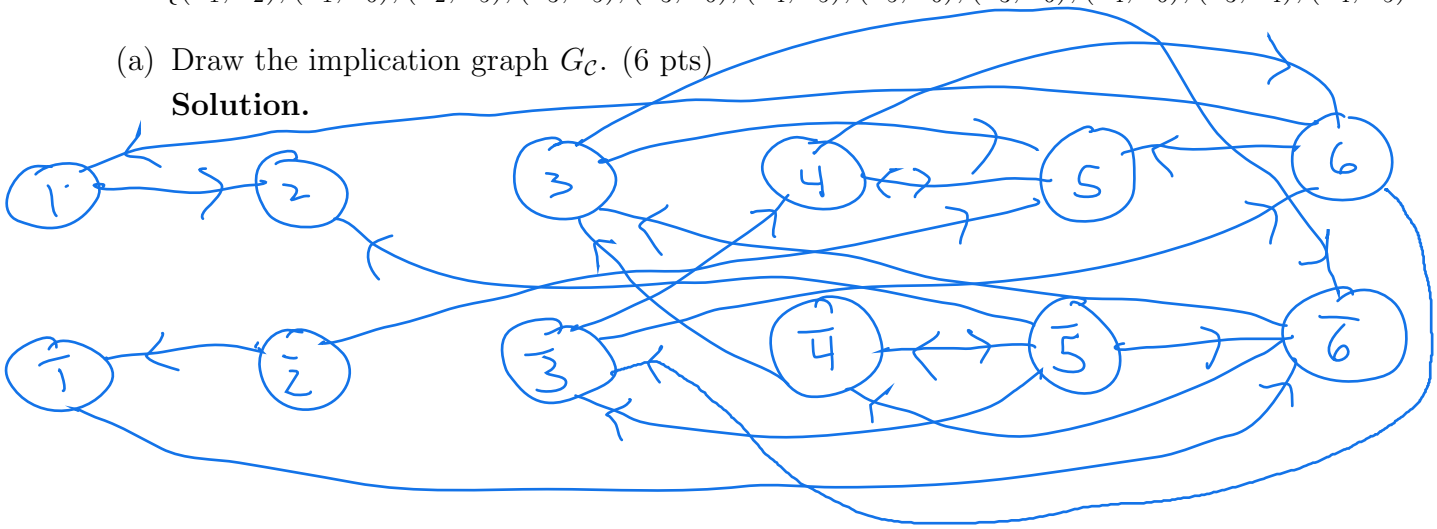
Solution. Answers may vary.

LO6. Draw the implication graph $G_{\mathcal{C}}$ associated with the `2SAT` instance

$$\mathcal{C} = \{(\overline{x}_1, x_2), (x_1, \overline{x}_6), (x_2, x_5), (\overline{x}_3, x_5), (x_3, x_6), (\overline{x}_4, x_5), (x_5, \overline{x}_6), (\overline{x}_3, \overline{x}_6), (\overline{x}_4, x_6), (x_3, x_4), (x_4, \overline{x}_5).$$

(a) Draw the implication graph $G_{\mathcal{C}}$. (6 pts)

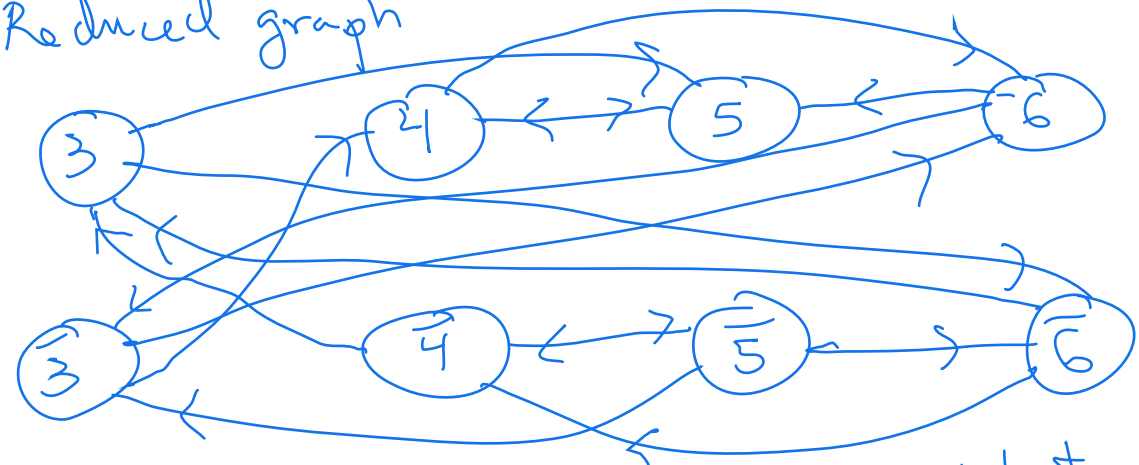
Solution.



(b) Perform the `Improved 2SAT` algorithm by computing the necessary reachability sets. Use numerical order (in terms of the variable index) and positive literal before negative literal when choosing the reachability set to compute next. Draw the resulting reduced `2SAT` instance whenever a consistent reachability set is computed. Either provide a final satisfying assignment for $\mathcal{C}$ or indicate why $\mathcal{C}$ is unsatisfiable. (12 pts)

$R(x_1) = \{x_1, x_2\} \Rightarrow \alpha_{R_{x_1}} = (x_1 = 1, x_2 = 1)$

Reduced graph



$R(x_3) = \{x_3, x_5, x_4, \bar{x}_6, \bar{x}_4\}$ is inconsistent

$R(\bar{x}_3) = \{\bar{x}_3, x_4, x_5, x_6\}$ is consistent $\alpha_{R_{\bar{x}_3}} = (x_3 = 0, x_4 = x_5 = x_6 = 1)$

$\alpha = (x_1 = x_2 = 1, x_3 = 0, x_4 = x_5 = x_6 = 1)$ satisfies $\mathcal{C}$

(c) Given satisfiable `2SAT` instance $\mathcal{C}$, if its implication graph $G_{\mathcal{C}}$ has the paths $P_1 = x_2, \overline{x}_4, \overline{x}_7, x_6, x_5$, and $P_2 = \overline{x}_7, \overline{x}_3, x_6, \overline{x}_2, x_8$, then what can you say about any satisfying assignment for $\mathcal{C}$. Justify your answer. (7 pts)

Solution. From P_1 there is the subpath $x_2, \overline{x}_4, \overline{x}_7$ and from P_2 there is the subpath $\overline{x}_7, \overline{x}_3, x_6, \overline{x}_2$. Concatenating these two paths yields a path from x_2 to $\overline{x}_2$, which means that x_2 cannot be assigned 1. Therefore, any satisfying assignment must assign x_2 the value 0.

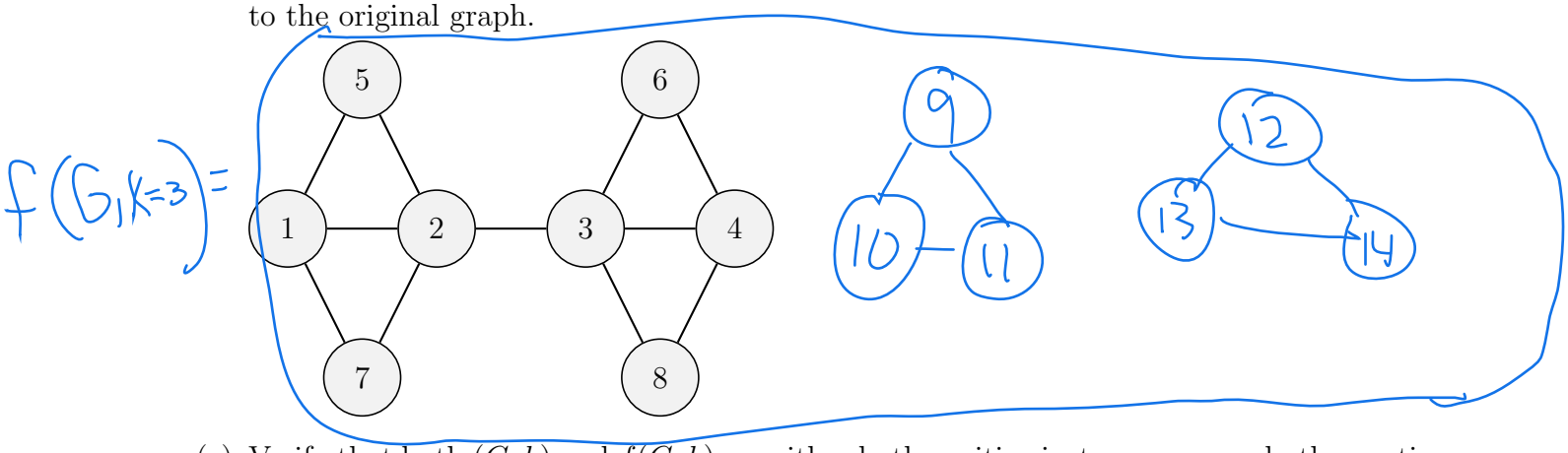
LO7. Answer the following.

(a) Provide the definition of what it means to be a mapping reduction from decision problem A to decision problem B. (5 pts)

Solution. See lecture notes.

(b) For the mapping reduction f : `Vertex Cover` $\rightarrow$ `Half Vertex Cover`, draw $f(G, k)$ for the `Vertex Cover` instance whose graph is shown below, and for which $k = 3$. (10 pts)

Solution. Since G has 8 vertices and $k = 3$, we must add $8 - 2(3) = 2$ isolated triangles to the original graph.



(c) Verify that both (G, k) and $f(G, k)$ are either both positive instances, or are both negative ones. Explain and show work. (10 pts)

Solution. Since both halves (left and right) of G require two vertices each to cover all the edges, it follows that G's smallest cover size is 4, and so $(G, k = 3)$ is a negative instance. Similarly, the image graph G' has a minimum cover size equal to $4 + 2(2) = 8$, since, in addition to the four vertices for the original G-edges, each of the added triangles requires two vertices to cover all its edges. Thus, G' is a negative instance of `HVC` since this graph has 14 vertices and does not admit a cover size equal to $14/2 = 7$. Therefore, the mapping is valid for instance $(G, k = 3)$. □

LO8. Do the following.

(a) An instance of `Matching` is a simple graph $G = (V, E)$ and a natural number $k \geq 1$. The problem is to decide if G has a **matching** of size k, i.e., a subset $M \subseteq E$ of k edges for no two edges in M share a vertex. Thus a natural certificate for `Matching` is a set $M \subseteq E$ of k edges. To show that `Matching` is an `NP` problem, do the following.

i. Provide the pseudocode for a verifier program for `Matching` that takes as inputs $G = (V, E)$, k, and M and returns 1 iff M is a matching for G. (8 pts)
Solution.

Initialize array used : $V \rightarrow \{0, 1\}$ so that used$(v) = 0$ for each $v \in V$.
For each $e = (u, v) \in M$,
  If used$(u) ==$ 1 $\vee$ used$(v) = 1$, then return 0.
  used$(u) = 1$. used$(v) = 1$.
Return 1.

ii. Given that the size parameters for `Matching` are $m = |E|$ and $n = |V|$, determine the big-O number steps required by your verifier. Defend your answer. (5 pts)
Solution. Initializing the `used` array requires $O(n)$ steps, while the program loop iterates $O(m)$ times and each iteration requires $O(1)$ steps in the form of array read/write operations. Therefore, the verifier requires $O(m + n)$ steps, proving that `Matching` $\in$ `NP`.

(b) Classify each of the following problems as being in `P`, `NP`, or co-`NP`. Note: at least three correct answers is necessary for passing this part of LO8. (3 points each).

i. An instance of `Total Distance` is a positively weighted directed graph $G = (\{1, 2, \ldots, n\}, E)$ and a natural number $k \geq 0$, and the problem is to decide if the entire sum of all the entries in matrix D does not exceed k, where entry $D_{ij} = 0$ if $i = 0$ or $j = 0$. Otherwise, D_{ij} equals the distance from vertex i to vertex j.

ii. An instance of `Balanced 3SAT` is a Boolean formula $F(x_1, \ldots, x_n)$. The problem is to decide if there are at least n^3 different variable assignments for which either every assignment satisfies F or every assignment does not satisfy F.

iii. An instance of `Weak Sets` is a simple graph $G = (V, E)$ and a natural number $k \geq 0$. The problem is to decide if for any subset $A \subseteq V$ having size k, there is always some vertex $v \notin A$ that is not adjacent to any of the vertices in A.

iv. An instance of `Boolean Formula Palindrome` is an array $a = F_1, \ldots, F_m$ of m Boolean formulas, each of which depends on the same set of variables $\{x_1, \ldots, x_n\}$. The problem is to decide if there is an assignment α over these n variables for which the Boolean array eval $= F_1(\alpha), \ldots, F_m(\alpha)$ of formula evaluations is a palindrome (i.e. reads the same both forwards and backwards).
Solution. P, P, co-NP, NP

Additional Problems

A1. Recall that an instance of the `3-Dimensional Matching (3DM)` decision problem consists of three sets A, B, and C, each having size n, along with a set S of triples of the form (a, b, c) where $a \in A$, $b \in B$, and $c \in C$. We assume that $|S| = m \geq n$. Also recall that an instance of the `MineSweep` decision problem is a simple graph $G = (V, E)$ and a map val : $V \to \mathcal{N}$ for which some of G's vertices are labeled with a positive integer. Namely, if val$(v) = 0$, then v is unlabeled, but if val$(v) > 0$, then val(v) gives the value of the label. The problem is to decide if there is a way to place mines on some of G's unlabeled vertices so that, for each $v \in V$ that is labeled with some integer $k =$ val(v), exactly k of v's neighbors have been assigned a mine. Do the following.

(a) Without using any specific examples, describe in the abstract an algorithm for map reducing an instance of (A, B, C, S) of `3DM` to an instance $(G = (V, E), \text{val} : V \to \mathcal{N})$ of `Mine Sweep`. How to construct G? Which vertices get assigned values and what are those values? These answers must directly depend on (A, B, C, S). (15 pts)
Solution. We have

$$V = A \cup B \cup C \cup S,$$

and each edge in E is of the form (u, t), where $u \in A \cup B \cup C$ and $t \in S$ is a triple for which one of t's components equals u. Finally, the only labeled vertices are those in $A \cup B \cup C$, with each being labeled with 1.

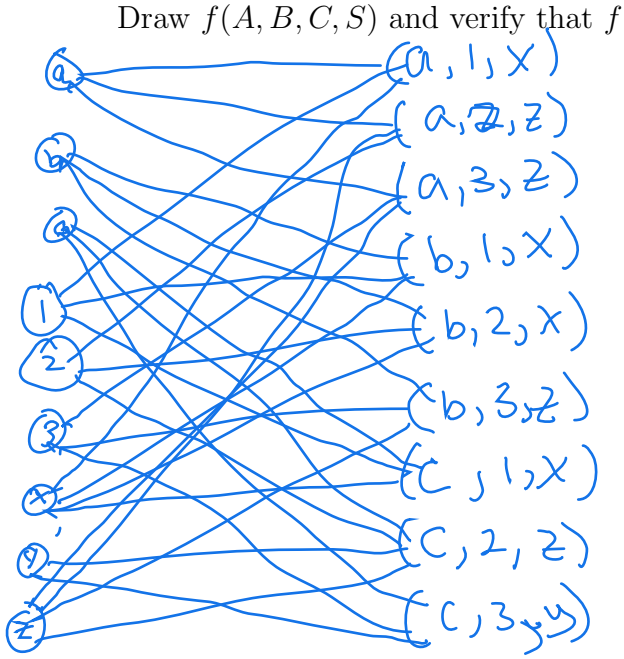
Now suppose T is a 3DM for (A, B, C, S). Then we may place mines on each member of T, leaving all other vertices without mines. Moreover, since T is a 3DM, for each $u \in A \cup B \cup C$, there is a unique $t \in T$ for which $u \in t$. Hence t is the only triple adjacent to u which has a mine placed on it. Therefore, $G = f(A, B, C, S)$ is a positive instance of `Mine Sweep`.

Conversely, if $G = (V, E, \text{label})$ is a positive instance of `Mine Sweep`. Then, since all members of $A \cup B \cup C$ are labeled and all members of S are unlabeled, it follows that the mines are placed only on members of S. Let $T \subseteq S$ denote the set of triples on which a mine has been placed. Then since each $u \in A \cup B \cup C$ is labeled with a 1, there is a unique tuple $t \in T$ that contains u. In other words, T is a 3DM and (A, B, C, S) is a positive instance of 3DM. □

(b) Apply your reduction f : `3DM` $\rightarrow$ `Mine Sweep` to the instance (A, B, C, S), where $A = \{a, b, c\}$, $B = \{1, 2, 3\}$, $C = \{x, y, z\}$, and

$$S = \{(a, 1, x), (a, 2, z), (a, 3, z), (b, 1, x), (b, 2, x), (b, 3, z), (c, 1, x), (c, 2, z), (c, 3, y)\}.$$

Draw $f(A, B, C, S)$ and verify that f is valid for this instance of `3DM`. (10 pts)



label(u) = 1 for all $u \in \{a, b, c, 1, 2, 3, x, y, z\}$

label(t) is undefined for all $t \in S$.

A2. Do the following.

(a) Recall the recurisve function $\text{mc}(i, A)$ that is defined for the `Runaway Traveling Salesperson (RTSP)` optimization problem, where positive integer i is the vertex of some weighted graph G, and A is a subset of the vertices of G. Provide a definition for what $\text{mc}(i, A)$ represents, and provide its recurrence. (10 pts)

Solution. See lecture notes (Solutions to the core dynamic programming exercises).

(b) A biproduct of computing the function $\text{mc}(i, A)$ for each of its possible inputs, is the computing of the function $\text{next}(i, A)$ (if you understand the meaning of `mc`, then you should also understand the meaning of `next` in relation to an instance of `RTSP` and its solution). Provide a *recursive* pseudocode implementation of the function

```
void print_path(int i, Set A, int[ ][ ] next)
```

which prints the optimal solution to the `RTSP` instance. You may assume access to the `print` function which accepts any integer and prints that integer surrounded by appropriate whitespace. (15 pts)

Solution.

`print`(i).
If $A == \emptyset$, then return.
$j = \text{next}[i][A]$.
`print_path`$(j, A - \{j\}, \text{next})$.

Makeup Problems

LO1. Solve the following problems.

(a) Use the Master Theorem to determine the growth of $T(n)$ if it satisfies the recurrence $T(n) = 81T(n/3) + n^{\log_2 18}$. Defend your answer.

(b) Use the substitution method to prove that, if $T(n)$ satisfies

$$T(n) = 4T(n/2) + 6\sqrt{n},$$

then $T(n) = \text{O}(n^2)$.

LO2. Solve the following problems.

(a) Use Strassen's products $P_1 = a(f - h) = af - ah$, $P_2 = (a + b)h = ah + bh$, $P_3 = (c + d)e = ce + de$, $P_4 = d(g - e) = dg - de$, $P_5 = (a + d)(e + h) = ae + ah + de + dh$, $P_6 = (b - d)(g + h) = bg + bh - dg - dh$, $P_7 = (a - c)(e + f) = ae - ce - cf + af$. to compute the matrix product

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} e & f \\ g & h \end{pmatrix}$$

Show all work.

(b) Draw the entire recursion tree that results when applying Mergesort to the array

$$a = 5, 4, 12, 8, 7, 11, 13, 9.$$

Label each node with the subproblem instance to be solved at that point of the recursion. Assume that a base case has size equal to one. Above each node in the tree, write the solution to the subproblem instance to which it pertains.

LO3. Do the following.

(a) Consider the FFT algorithm when applied to a polynomial $A(x)$ having degree $2^n - 1$. Provide the equation that relates $A(x)$ to the two subproblem polynomials $A_e(x)$ and $A_o(x)$. What are the degrees of these two polynomials? Based on this equation, why is it essential that, for even n, the nth roots of unity come in additive-inverse pairs?

(b) If $p(x) = 3 - 2x + 1x^2 - 5x^3$, then compute $\text{DFT}^{-1}(p)$ using the IFFT algorithm. Show the entire recursion tree as was done in the lecture notes.

LO4. Do the following.

(a) Recall the use of the disjoint-set data structure for the purpose of improving the running time of the `Unit Task Scheduling` algorithm. For the set of tasks

Task	a	b	c	d	e	f
Deadline Index	4	5	4	4	3	5
Profit	60	50	40	30	20	10

show the M-Tree forest after it has been inserted (or at least has attempted to be inserted in case the scheduling array is full). Note that the earliest possible deadline index is 1, meaning that the earliest slot in the schedule array has index 1. Also, assume that an insert attempt that takes place at index i results in the function call `root`(i), followed by a `union` operation. Finally, to receive credit, your solution should show six different snapshots of the M-Tree forest.

(b) For the greedy algorithm that solves the **Fuel Reloading Problem**, in *one sentence* describe the greedy choice that is made in each round of the algorithm. Do *not* provide an entire description of the algorithm.

(c) Given the station locations

$$10, 13, 19, 23, 26, 32, 36, 47, 56, 66, 73, 77, 89, 98,$$

determine the *least* distance d that a car needs to be able to travel on a full tank of fuel in order to be able to reach location 100 starting from location 0. For this distance, provide a minimum set of stations that it must visit.